Myanmar Institute of Information Technology

Faculty of Computer Systems and Technologies

CSE 3050 Assignment

1. Show how to use the 10-member instruction set of Table 1 to implement the following operations that correspond to single instructions in many computers; use as few instructions as you can. (a) Copy the contents of memory location X to memory location Y. (b) Increment the accumulator AC. (c) Branch to a specified address adr if AC ^ 0.

Table 1

Type	Instruction	HDL format	Assembly-language format	Narrative format (comment)
Data transfer	Load	AC := M(*X*)	LD X	Load *X* from M into AC.
	Store	M(*X*) := AC	ST X	Store contents of AC in M as *X*.
	Move register	DR := AC	MOV DR, AC	Copy contents of AC to DR.
	Move register	AC := DR	MOV AC, DR	Copy contents of DR to AC.
Data processing	Add	AC := AC + DR	ADD	Add DR to AC.
	Subtract	AC := AC − DR	SUB	Subtract DR from AC.
	And	AC := AC *and* DR	AND	And bitwise DR to AC.
	Not	AC := *not* AC	NOT	Complement contents of AC
Program control	Branch	PC := M(*adr*)	BRA adr	Jump to instruction with address *adr*.
	Branch zero	if AC = 0 then PC := M(*adr*)	BZ adr	Jump to instruction *adr* if AC = 0.

2. Use the instruction set of Table 1 to implement the following two operations assuming that sign-magnitude code is used, (a) AC := -M(X). (b) Test the right-most bit b of the word stored in a designated memory location X. If b = 1, clear AC; otherwise, leave AC unchanged. [Hint: Use an AND instruction to mask out certain bits of a word.

Line	Location	Instruction or data	Comment
0	*one*	00 . . . 001	The constant one.
1	*mult*	*N*	The multiplier.
2	*ac*	00 . . . 000	Location for initial value *Y* of AC.
3	*prod*	00 . . . 000	Location for (partial) product *P*.
4		ST *ac*	Save initial value *Y* of AC.
5	*loop*	LD *mult*	Load *N* into AC to test for termination.
6		BZ *exit*	Exit if *N* = 0; otherwise continue.
7		LD *one*	Load 1 into AC.
8		MOV DR, AC	Move 1 from AC to DR.
9		LD *mult*	Load *N* into AC to decrement it.
10		SUB	Subtract 1 from *N*.
11		ST *mult*	Store decremented *N*.
12		LD *ac*	Load initial value *Y* of AC.
13		MOV DR, AC	Move *Y* from AC to DR.
14		LD *prod*	Load current partial product *P*.
15		ADD	Add *Y* to *P*.
16		ST *prod*	Store the new partial product *P*.
17		BRA *loop*	Branch to *loop*.
18	*exit*	. . .	

Figure 1

3. Consider the possibility of overlapping instruction fetch and execute operations when executing the multiplication program of Figure 1. (a) Assuming only one word can be transferred over the system bus at a time, determine which instructions can be over-lapped with neighboring instructions, (b) Suppose that the

Myanmar Institute of Information Technology

Faculty of Computer Systems and Technologies

CSE 3050 Assignment

CPU-memory interface is redesigned to allow one instruction fetch and one data load or store to occur during the same clock cycle. Now determine which instructions, if any, in the multiplication can-not be overlapped with neighboring instructions.

4. Write a brief note discussing one advantage and one disadvantage of each of the following two unusual features of the ARM6: (a) the inclusion of the program counter PC in the general register file; (b) the fact that execution of every instruction is conditional.

5. Use HDL notation and ordinary English to describe the actions performed by each of the following ARM6 instructions: (a) MOV R6, #0; (b) MVN R6, #0; (c) ADD R6, R6, R6; (d) EOR R6, R6, R6.

6. Suppose the ARM6 has the following initial register contents (all given in hex code):

Rl = 11110000; R2 = 0000FFFF; R3 = 12345678; NZCV = 0000

Identify the new contents of every register or flag that is changed by execution of the following instructions. Assume each is executed separately with the foregoing initial state, (a) MOV R1, R2; (b) MOVCS R1. R2; (c) MVNCS R2. R1; (d) MOV R3, #0;(e) MOV R3, R4, LSL#4.

7. Suppose the ARM6 has the following initial register and memory contents (all given in hex code):

Rl = 00000000; R2 = 87654321; R3 = A05B77F9; NZCV = 0000

Identify the new contents of every register or flag that is changed by execution of the following instructions. Assume each is executed separately with the foregoing initial state, (a) ADD R1, R2, R3; (b) ADDS R1. R3.R3; (c) SUBS R2,R1,#1; (d) ANDS R3, R2, R1 ; (e) EORCSS R1, R2, R3

8. Use the instruction set for the ARM6 given in Figure 2 to write short code segments to perform the tasks given below. Note that an opcode can be followed by two optional suffixes, a two-character condition code to determine branching and S to activate the status flags. Figure 4 lists all possible condition fields. The required tasks are:(a) Replace the contents of register Rl by its absolute value, (b) Perform the 64-bit subtraction R5.R4 := R1.R0 - R3.R2, where the even-numbered registers contain the right (less significant) half of each operand.

9. Write the shortest ARM6 program that you can to implement the following conditional statement:

while ($x \neq y$) **do** x := y – 1;

Assume that x and y are stored in CPU registers R1 and R2, respectively

10. Identify five major differences between the instruction sets of the ARM6 and the680X0 and comment on their impact on the CPU cost and performance.

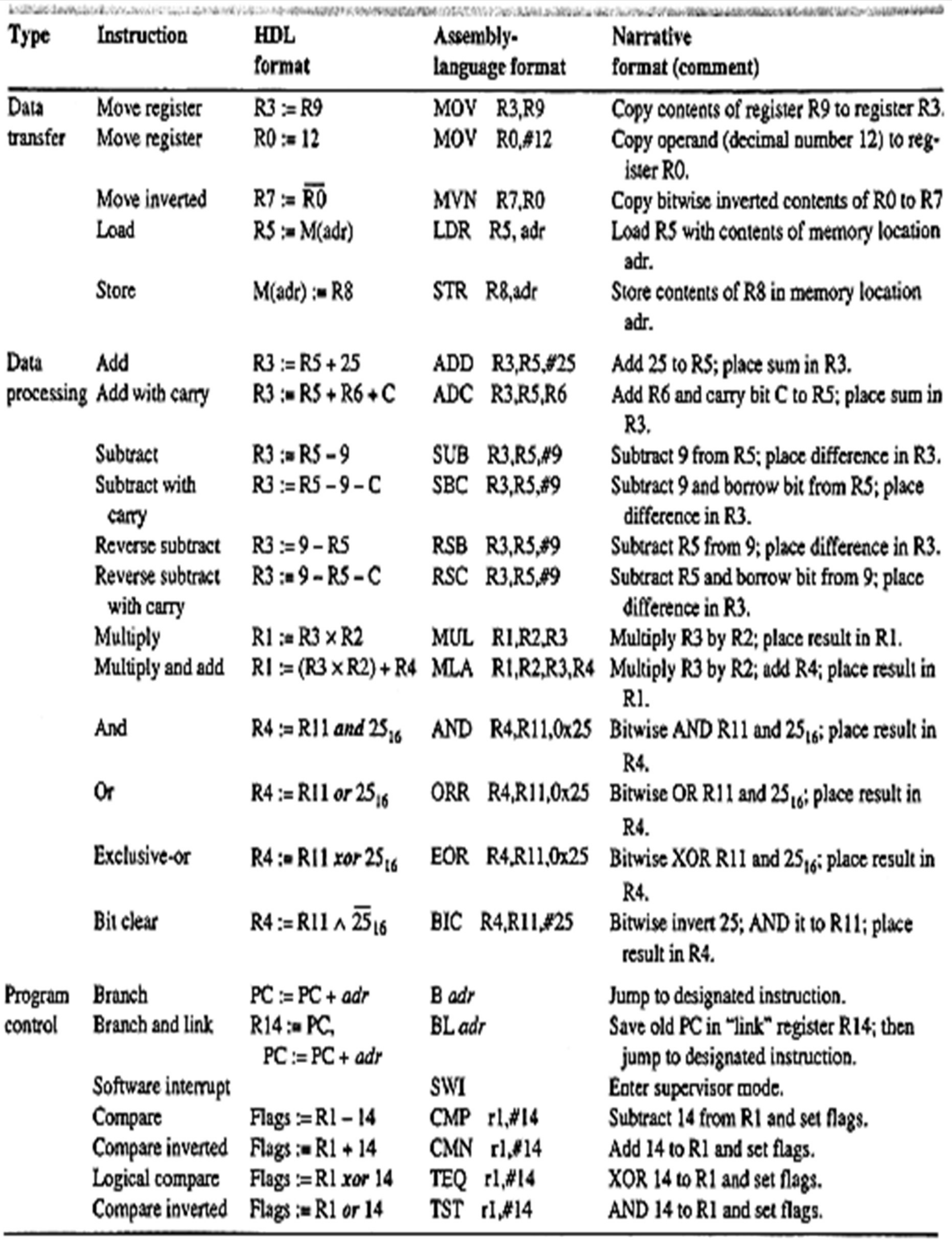
11. Use HDL notation and ordinary English to write the actions performed by each of the following 680X0 instructions: (a) MOVE (A5)+, D5; (b) ADD.B $2A10,D0;(c) SUBI #10,(A0); (d) AND.L #SFF,D0.

12. The 680X0 has two types of unconditional branch instructions BRA (branch always) and JMP (jump). Therefore, branch to statement L can be implemented either by BRAL or JMP L. What is the difference between these two instructions? Under what circumstances is each type of branch instruction preferred?

Myanmar Institute of Information Technology

Faculty of Computer Systems and Technologies

CSE 3050 Assignment



Type	Instruction	HDL format	Assembly-language format	Narrative format (comment)
Data transfer	Move register	R3 := R9	MOV R3,R9	Copy contents of register R9 to register R3.
	Move register	R0 := 12	MOV R0,#12	Copy operand (decimal number 12) to register R0.
	Move inverted	R7 := $\overline{\text{R0}}$	MVN R7,R0	Copy bitwise inverted contents of R0 to R7
	Load	R5 := M(adr)	LDR R5, adr	Load R5 with contents of memory location adr.
	Store	M(adr) := R8	STR R8,adr	Store contents of R8 in memory location adr.
Data processing	Add	R3 := R5 + 25	ADD R3,R5,#25	Add 25 to R5; place sum in R3.
	Add with carry	R3 := R5 + R6 + C	ADC R3,R5,R6	Add R6 and carry bit C to R5; place sum in R3.
	Subtract	R3 := R5 − 9	SUB R3,R5,#9	Subtract 9 from R5; place difference in R3.
	Subtract with carry	R3 := R5 − 9 − C	SBC R3,R5,#9	Subtract 9 and borrow bit from R5; place difference in R3.
	Reverse subtract	R3 := 9 − R5	RSB R3,R5,#9	Subtract R5 from 9; place difference in R3.
	Reverse subtract with carry	R3 := 9 − R5 − C	RSC R3,R5,#9	Subtract R5 and borrow bit from 9; place difference in R3.
	Multiply	R1 := R3 × R2	MUL R1,R2,R3	Multiply R3 by R2; place result in R1.
	Multiply and add	R1 := (R3 × R2) + R4	MLA R1,R2,R3,R4	Multiply R3 by R2; add R4; place result in R1.
	And	R4 := R11 *and* 25_{16}	AND R4,R11,0x25	Bitwise AND R11 and 25_{16}; place result in R4.
	Or	R4 := R11 *or* 25_{16}	ORR R4,R11,0x25	Bitwise OR R11 and 25_{16}; place result in R4.
	Exclusive-or	R4 := R11 *xor* 25_{16}	EOR R4,R11,0x25	Bitwise XOR R11 and 25_{16}; place result in R4.
	Bit clear	R4 := R11 $\wedge$ $\overline{25}_{16}$	BIC R4,R11,#25	Bitwise invert 25; AND it to R11; place result in R4.
Program control	Branch	PC := PC + *adr*	B *adr*	Jump to designated instruction.
	Branch and link	R14 := PC, PC := PC + *adr*	BL *adr*	Save old PC in "link" register R14; then jump to designated instruction.
	Software interrupt		SWI	Enter supervisor mode.
	Compare	Flags := R1 − 14	CMP r1,#14	Subtract 14 from R1 and set flags.
	Compare inverted	Flags := R1 + 14	CMN r1,#14	Add 14 to R1 and set flags.
	Logical compare	Flags := R1 *xor* 14	TEQ r1,#14	XOR 14 to R1 and set flags.
	Compare inverted	Flags := R1 *or* 14	TST r1,#14	AND 14 to R1 and set flags.

Figure 2

Myanmar Institute of Information Technology

Faculty of Computer Systems and Technologies

CSE 3050 Assignment

Code	Mnemonic	Flag test	Usual interpretation
0000	EQ	$Z = 1$	Result equal to zero.
0001	NE	$Z = 0$	Result not equal to zero.
0010	CS or HS	$C = 1$	Unsigned overflow: result higher or same.
0011	CC or LO	$C = 0$	No unsigned overflow: result lower.
0100	MI	$N = 1$	Result negative.
0101	PL	$N = 0$	Result positive or zero.
0110	VS	$V = 1$	Signed overflow.
0111	VC	$V = 0$	No signed overflow.
1000	HI	$C = 1$ and $Z = 0$	Unsigned result higher.
1001	LS	$C = 0$ or $Z = 1$	Unsigned result lower or same.
1010	GE	$N = V$	Signed result greater or equal.
1011	LT	$\bar{N} = V$	Signed result less than.
1100	GT	$Z = 0$ and $N = V$	Signed result greater than.
1101	LE	$Z = 1$ or $\bar{N} = V$	Signed result less than or equal.
1110	AL	None	Always (unconditional branch).
1111	NV	None	Never (no branching)

Figure 4: Conditional Codes of the ARM6 and their Interpretations

13. Write a program for the 680X0 that replaces the word DATA stored in memory location ADR by its bitwise logical complement $\overline{DATA}$ if and only if DATA $\neq 0$.

14. Modify the vector addition program of Figure 6 (Example in Lecture) to compute the sum C := A + B for 100 instead of 1000 one-byte decimal numbers. Assume that the locations of the A and B operands are unchanged, but the result C is now required to replace(overwrite) B.

Location	Instruction		Comment
	MOVE.L	#2001, A0	Load address 2001 into register A0 (pointer to vector A).
	MOVE.L	#3001, A1	Load address 3001 into register A1 (pointer to vector B).
	MOVE.L	#4001, A2	Load address 4001 into register A2 (pointer to vector C).
START	ABCD	-(A0), -(A1)	Decrement contents of A0 and A1 by 1, then add M(A0) to M(A1) using 1-byte decimal addition.
	MOVE.B	(A1), -(A2)	Decrement A2 and then store the 1-byte sum M(A1) in location M(A2) of vector C.
TEST	CMPA	#1001, A0	Compare 1001 to address in A0. If equal, set the Z flag (condition code) to 1; otherwise, reset Z to 0.
	BNE	START	Branch to START if Z is not equal to 1.

Figure 6

Myanmar Institute of Information Technology

Faculty of Computer Systems and Technologies

CSE 3050 Assignment

15. Suppose that the hex contents of two CPU registers in a 32-bit processor are as follows:

R0 = 01237654:   Rl = 7654EDCB

The following store-word instructions are executed to transfer the contents of these registers to main memory M.

STORE R0, ADR

STORE Rl.ADR+4

Assuming that M is byte-addressable, give the contents of all memory locations affected by the above code (a) if the computer is big-endian and (b) if the computer is little- endian.

16. Suppose that in the 6-bit floating-point format illustrated by Figure 5, B = 2, E is a3-bit sign-magnitude integer as before, but M is now a 3-bit sign-magnitude fraction. (a) What are the decimal values of the largest and smallest nonzero real numbers that can be represented by this format? (b) How many different real numbers can be represented?

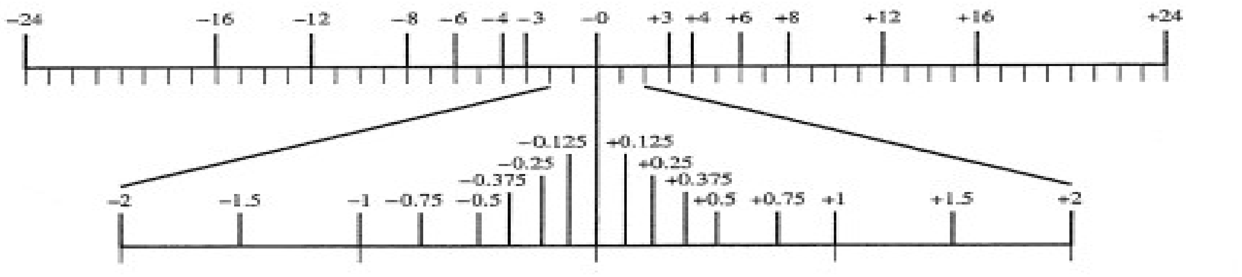


Figure 5

17. Obtain the (approximate) decimal values that conform to the IEEE 754 floating-point format of the following two numbers:

A= 100101111 10000000000000000000000

B = 0 10001110 00000000000000000000001

18. Derive the correct floating-point representation for the decimal numbers +3.25 and-3.25 using the 32-bit IEEE 754 floating-point standard.

19. Consider a 32-bit RISC-style processor P whose only addressing modes for register-to-register instructions are immediate and direct and whose only addressing mode for load/store instructions is register indirect with offset. Assume also that the CPU has 64general-purpose registers R0:R63 that can serve either as data or address registers. A single 32-bit instruction format contains four fields: an opcode, two register fields, and a 16-bit immediate address field, (a) What is the maximum number of opcode types? (b) Using an ad hoc but typical assembly-language notation with clear comments, de-scribe how a single instruction of P might perform each of the following three operations: load a word from M; store a byte in M; double the number word stored in a register (there is no multiply opcode).